What is claimed is:

1. A computer language translator for translating a first computer language source code to a second computer language source code, comprising:

a computer;

a library accessible by said computer, said library including data indicative of types of data manipulations between the first computer language source code and the second computer language source code;

software executing on said computer, said software analyzing the first computer language source code to identify a type of data manipulation that the first computer language source code performs, accessing said library and correlating the type of data manipulation the first computer language source code performs to second computer language source code, the correlation being independent of the context in which the first computer language source code is used, said software generating second computer language source code that emulates the type of data manipulation the first computer language source code performs.

2. The computer language translator according to claim 1 wherein the translator is a bi-directional translator, said software analyzing the second computer language source code to identify the type of data manipulation that the second computer language source code performs, accessing said library and correlating the type of data manipulation the second computer language source code performs to first computer language source code, the correlation being independent of the context in which the second computer language source code is used, said software generating re-translated first computer language source code that emulates the type of data manipulation the second computer language source code performs.

3. The computer language translator according to claim 2 wherein the re-translated first computer language source code is substantially identical to the first computer language source code.

4. The computer language translator according to claim 1 wherein said software performs a name adjustment when an incompatibility between the first computer language source code and the second computer language source code occurs during correlation of the type of data manipulation the first computer language source code performs to the second computer language source code to resolve the incompatibility, the name adjustment being independent of the context in which the incompatibility is located.

5. The computer language translator according to claim 1 wherein the first computer language source code comprises a class.

6. The computer language translator according to claim 5 wherein the class consists of units selected from the group consisting of: methods, data fields, inner-classes, and combination thereof.

7. The computer language translator according to claim 1 wherein the first computer language source code comprises an identifier.

8. The computer language translator according to claim 7 wherein said software performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during correlation of the type of data manipulation the identifier performs to the second computer language source code to resolve the incompatibility, the name adjustment being independent of the context in which the identifier is used.

9. The computer language translator according to claim 1 wherein said software generates a tagged element inserted in the second computer language source code indicative of a type of data manipulation the first computer language source code performs.

10. The computer language translator according to claim 9 wherein the tagged element comprises information selected from the group consisting of: formatting, translation data, and first computer language source code.

11. The computer language translator according to claim 1 wherein the first computer language is Java and the second computer language is C++.

12. The computer language translator according to claim 1 wherein the first computer language is Java and the second computer language is C#.

13. The computer language translator according to claim 1 wherein the first computer language is C# and the second computer language is C++.

14. A computer language translating system for translating a first computer language source code to a second computer language source code, comprising:

a library accessible by said translating system, said library including data indicative of types of data manipulations between the first computer language source code and the second computer language source code;

a parser for parsing the first computer language source code into parsed elements;

an analyzer for analyzing the parsed elements to identify the type of data manipulation that the first computer language source code performs, said analyzer accessing said library and correlating the type of data manipulation the first computer language source code performs to second computer language source code, the correlation being independent of the context in which the first computer language source code is used; and

a generator for generating second computer language source code that emulates the type of data manipulation the first computer language source code performs.

15. The computer language translating system according to claim 14 wherein the translating system is a bi-directional translator, said parser parsing the second computer language source code into parsed elements, said analyzer analyzing the parsed elements to identify the type of data manipulation that the second computer language source code performs, said analyzer accessing said library and correlating the type of data manipulation the second computer language source code performs to first computer language source code, the correlation being independent of the context in which the second computer language source code is used, and said generator generating re-translated first computer language source code that emulates the type of data manipulation the second computer language source code performs.

16. The computer language translating system according to claim 15 wherein the re-translated first computer language source code is substantially identical to the first computer language source code.

17. The computer language translating system according to claim 14 wherein said generator performs a name adjustment when an incompatibility between the first computer language source code and the second computer language source code occurs during generation of the second computer language source code, the name adjustment being independent of the context in which the incompatibility is located.

18. The computer language translating system according to claim 14 wherein the first computer language source code comprises an identifier.

19. The computer language translating system according to claim 18 wherein said generator performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during correlation of the type of data manipulation the first computer language source code performs to second computer language source code, the name adjustment being independent of the context in which the identifier is used.

20. The computer language translating system according to claim 14 wherein said generator generates a tagged element inserted in the second computer language source code indicative of the type of data manipulation the first computer language source code performs.

21. The computer language translating system according to claim 20 wherein the tagged element comprises information selected from the group consisting of: formatting, translation data, and first computer language source code.

22. The computer language translating system according to claim 14 wherein the first computer language is Java and the second computer language is C++.

23. The computer language translating system according to claim 14 wherein the first computer language is Java and the second computer language is C#.

24. The computer language translating system according to claim 14 wherein the first computer language is C# and the second computer language is C++.

25. A method for translating a first computer language source code to a second computer language source code comprising the steps of:

inputting the first computer language source code into a computer;

parsing the first computer language source code into parsed elements;

analyzing the parsed elements to determine a type of data manipulation the first computer language source code performs;

building class declarations and class definitions from the parsed elements that are independent of the context of the first computer language source code;

generating the second computer language source code according to the class declarations and class definitions, the second computer language source code emulating the type of data manipulation the first computer language source code performs.

26. The translation method according to claim 25 further comprising the steps of:

parsing the second computer language source code into parsed elements;

analyzing the parsed elements to determine a type of data manipulation the second computer language source code performs;

building class declarations and class definitions from the parsed elements that are independent of the context of the second computer language source code;

generating the re-translated first computer language source code according to the class declarations and class definitions, the first computer language source code emulating the type of data manipulation the second computer language source code performs.

27. The translation method according to claim 25 further comprising the step of, performing a name adjustment when an incompatibility between the first computer language source code and the second computer language source code occurs during translation, the name adjustment being independent of the context in which the incompatibility is located.

28. The translation method according to claim 25 wherein the first computer language source code comprises an identifier.

29. The translation method according to claim 28 further comprising the step of, performing a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during translation, the name adjustment being independent of the context in which the identifier is located.

30. The translation method according to claim 25 further comprising the step of,

generating a tagged element indicative of the type of data manipulation the first computer language source code performs; and

inserting the tagged element in the second computer language source code.

31. The translation method according to claim 25 wherein the first computer language is Java and the second computer language is C++.

32. The computer language translating system according to claim 25 wherein the first computer language is Java and the second computer language is C#.

33. The translation method according to claim 25 wherein the first computer language is C# and the second computer language is C++.

34. A method for translating a first computer language source code to a second computer language source code comprising the steps of:

analyzing the first computer language source code to determine a type of data manipulation the first computer language source code performs;

correlating the type of data manipulation of the first computer language source code performs to second computer language source code, the correlation being independent of the context in which the first computer language source code is used; and

generating the second computer language source code, the second computer language source code emulating the type of data manipulation the first computer language source code performs.

35. The translation method according to claim 34 further comprising the steps of:

analyzing the second computer language source code to determine a type of data manipulation the second computer language source code performs;

correlating the type of data manipulation of the second computer language source code performs to first computer language source code, the correlation being independent of the context in which the second computer language source code is used; and

generating re-translated first computer language source code, the re-translated first computer language source code emulating the type of data manipulation the first computer language source code performs, and being substantially identical to the first computer language source code.

36. The translation method according to claim 34 further comprising the step of, performing a name adjustment when an incompatibility between the first computer language source code and the second computer language

source code occurs during translation, the name adjustment being independent of the context in which the incompatibility is used.

37. The translation method according to claim 34 wherein the first computer language source code comprises an identifier.

38. The translation method according to claim 36 further comprising the step of, performing a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during translation, the name adjustment being independent of the context in which the identifier is located.

39. The translation method according to claim 34 further comprising the step of,

generating a tagged element indicative of the type of data manipulation the first computer language source code performs; and

inserting the tagged element in the second computer language source code.

40. The translation method according to claim 34 wherein the first computer language is Java and the second computer language is C++.

41. The translation method according to claim 34 wherein the first computer language is Java and the second computer language is C#.

42. The translation method according to claim 34 wherein the first computer language is C# and the second computer language is C++.

43. A computer language translating system for translating a Java source code to an OOP language source code, comprising:

a library including data indicative of types of data manipulations between the Java source code and the OOP language source code;

an analyzer for analyzing the Java source code to determine a type of data manipulation the Java source code performs, said analyzer accessing said library to correlate the type of data manipulation the Java source code performs to OOP language source code, the correlation being independent of the context in which the Java source code is used; and

a generator for generating OOP language source code that emulates the type of data manipulation the Java source code performs.

44. The computer language translating system according to claim 43 wherein said analyzer analyzes the OOP language source code to determine a type of data manipulation the OOP language source code performs, said analyzer accessing said library to correlate the type of data manipulation the OOP language source code performs to Java source code, the correlation being independent of the context in which the OOP language source code is used, and said generator generates re-translated Java source code that emulates the type of data manipulation the OOP language source code performs, the re-translated Java source code being substantially identical to the Java source code.

45. The computer language translating system according to claim 43 wherein said analyzer builds class declarations and class definitions based on the type of data manipulation the Java source code performs and said generator generates OOP language source code based upon the class declarations and class definitions.

46. A method for translating a portion of Java source code that comprises at least one class to an OOP language source code comprising the steps of:

parsing the at least one class into multiple identifiers;

parsing one of the identifiers into name segments;

adjusting the name segments individually and independently from each other according to the OOP language source code;

generating OOP language source code based on the name segments where only a portion of the Java source code is translated to the OOP language source code, the OOP language source code emulating a type of data manipulation the portion of the Java source code performs.

47. A method for translating a first computer language source code that performs a data manipulation with a virtual machine to a second computer language source code that performs a data manipulation without use of a virtual machine, comprising the steps of:

analyzing the first computer language source code to determine a type of data manipulation the first computer language source code performs;

correlating the type of data manipulation of the first computer language source code performs to second computer language source code;

generating the second computer language source code, the second computer language source code emulating the type of data manipulation the first computer language source code performs; and

performing a data manipulation with the second computer language source code that emulates the type of data manipulation the first computer language source code performs, the second computer language source code performing the data manipulation without use of a virtual machine.

48. A method for translating a first computer language source code that performs a data manipulation with a garbage collector to a second computer language source code that performs a data manipulation without use of a garbage collector, comprising the steps of:

analyzing the first computer language source code to determine a type of data manipulation the first computer language source code performs;

correlating the type of data manipulation of the first computer language source code performs to second computer language source code;

generating the second computer language source code, the second computer language source code emulating the type of data manipulation the first computer language source code performs; and

performing a data manipulation with the second computer language source code that emulates the type of data manipulation the first computer language source code performs, the second computer language source code performing the data manipulation without use of a garbage collector.